ORACLE

OCI Exports metrics to Prometheus Solution

Jingsong.liu@oracle.com
2022 年 4 月 18, Version 1.0

Table of contents

产品版本

OCI 云平台

prometheus-2.43.0.windows-amd64

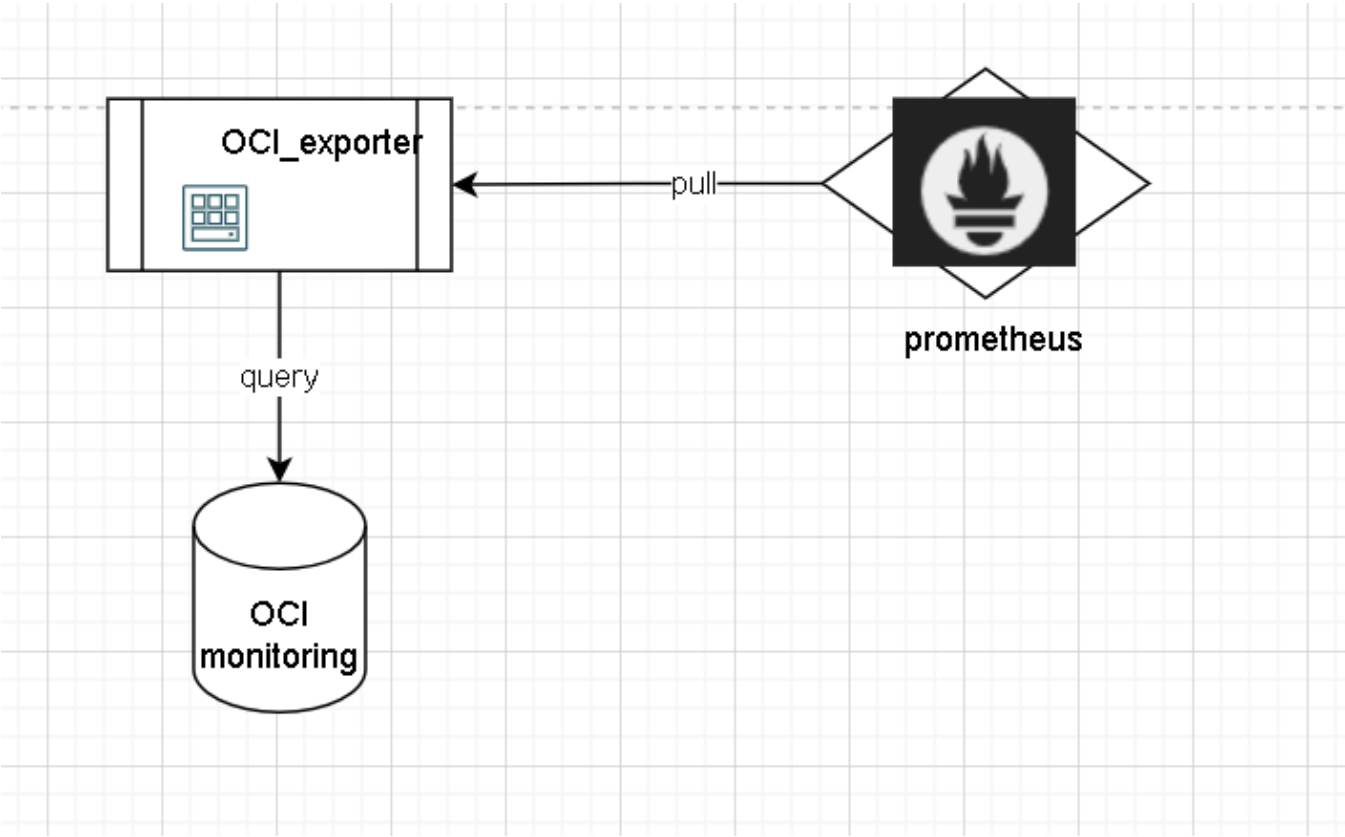
Golang 1.20, go.mod as below

```
github.com/oracle/oci-go-sdk/v65 v65.36.0
github.com/prometheus/client_golang v1.15.0
```

Overview

Prometheus is a very popular monitoring solution in industry, so we need to have the requirement that exports metrics to Prometheus.

About the usage we only need to Get the binary and make metrics.yaml file Prepared and then run the application In a virtual machine. Let the Prometheus discover These instances of exporters. then the metrics will be exported to Prometheus



The binary of windows and linux can be downloaded from github

https://github.com/munger1985/OCI-Auto-Scripts/tree/main/oci_exporter

For usage, we can simply add metrics in metrics.yaml

```
---
metrics:
  - name: oci_all_lb_bytesent
    help: "The number of bytes sent from all load balancer."
    type: many
    interval: 3
    namespace: oci_lbaas
    mql: BytesSent[1m].mean()
    compartmentId: ocid1.compartment.oc1..aaaaaaaahr7aicqtodxmcfor6pbqn3hvsngpftozyxzqw36gj4kh3w3kkj4q
    label1: dev
    label2: high
  - name: oci_vm1_mem
```

```
  help: "vm1 mem usage."
  type: single
  interval: 1
  namespace: oci_computeagent
  mql: MemoryUtilization[1m]{resourceId = "ocid1.instance.oc1.ap-singapore-
1.anzwsljrak7gbriccijo42npujwjrub3j6e6fkoe7ijtpcn4krhndgr3f2da"}.mean()
  compartmentId: ocid1.compartment.oc1..aaaaaaaahr7aicqtodxmcfor6pbqn3hvsngpftozyxzqw36gj4kh3w3kkj4q
  label1: dev
  label2: high
# - name: oci_vm_mem
#   help: "oci mem usgae ."
#   type: single
#   interval: 1
#   namespace: oci_computeagent
#   mql: DisklopsWritten[1m]{resourceId = "ocid1.instance.oc1.ap-singapore-
1.anzwsljrak7gbriccijo42npujwjrub3j6e6fkoe7ijtpcn4krhndgr3f2da"}.rate()
#   label:
#     a: b
```

You can comment some metric whenever you want

Field explanation

Name: name of the metric

Help: description

Type: if this metric will get you many datapoints, then it is many, otherwise single, many is used to query many resource in one go, you can all use many.

Interval: export query data in this interval, unit: minute

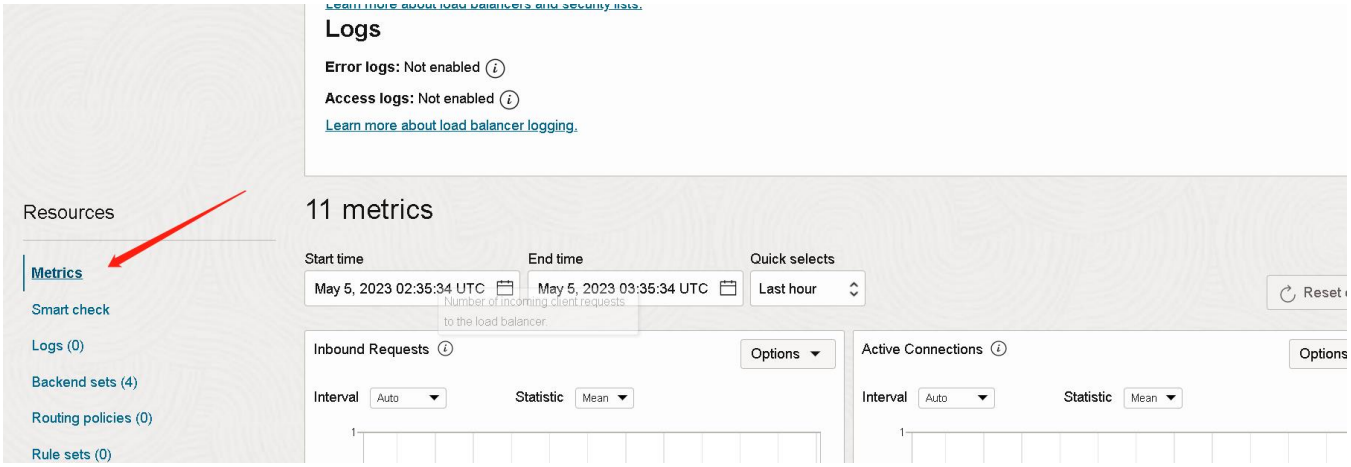
Namespace: for query data

compartmentId: compartment of the target resource

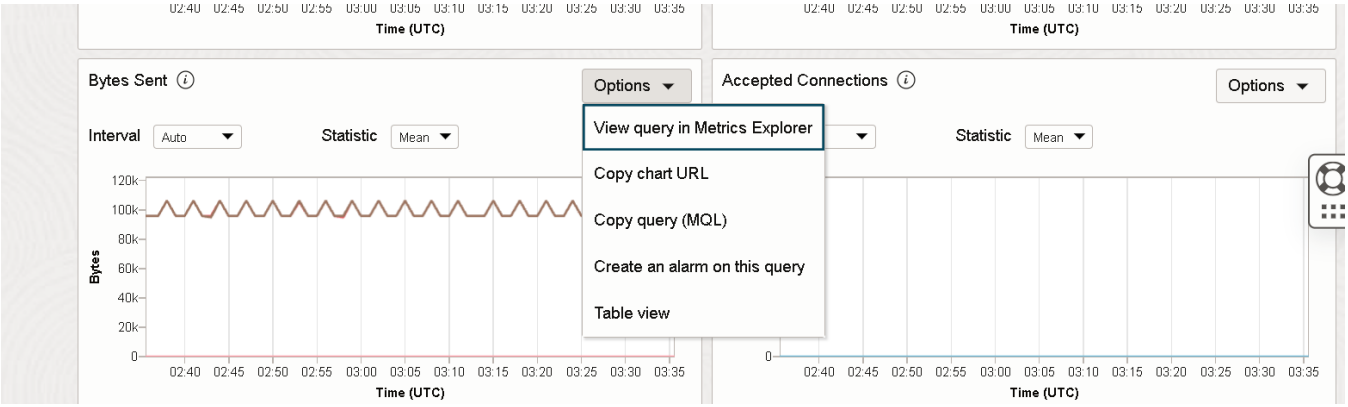
Label1: value of custom label, can be used for Prometheus filter

Label2: value of custom label, can be used for Prometheus filter

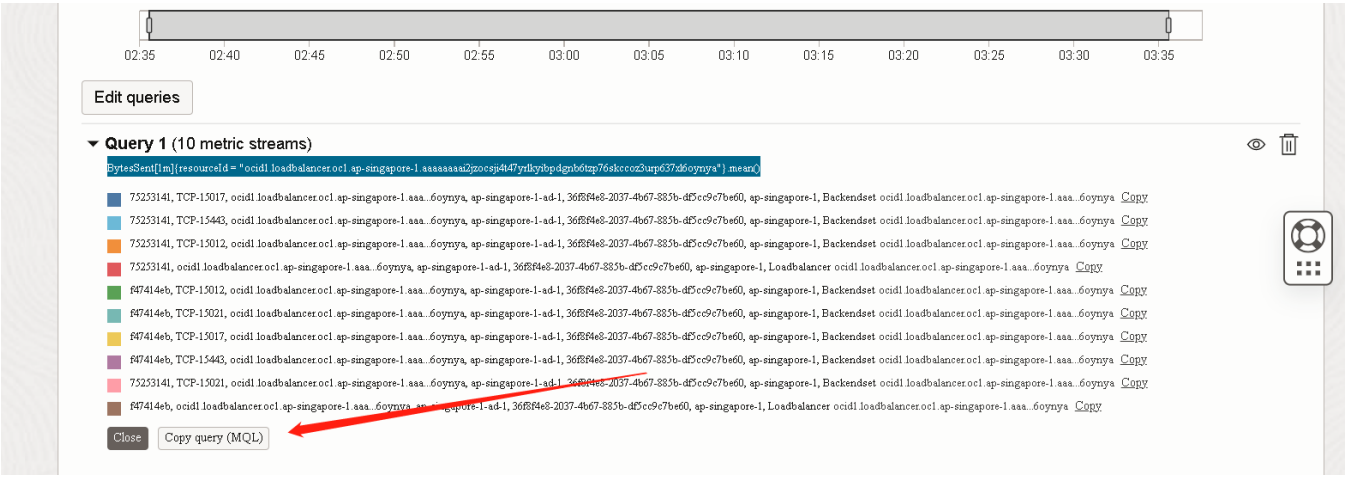
How to get namespace and MQL

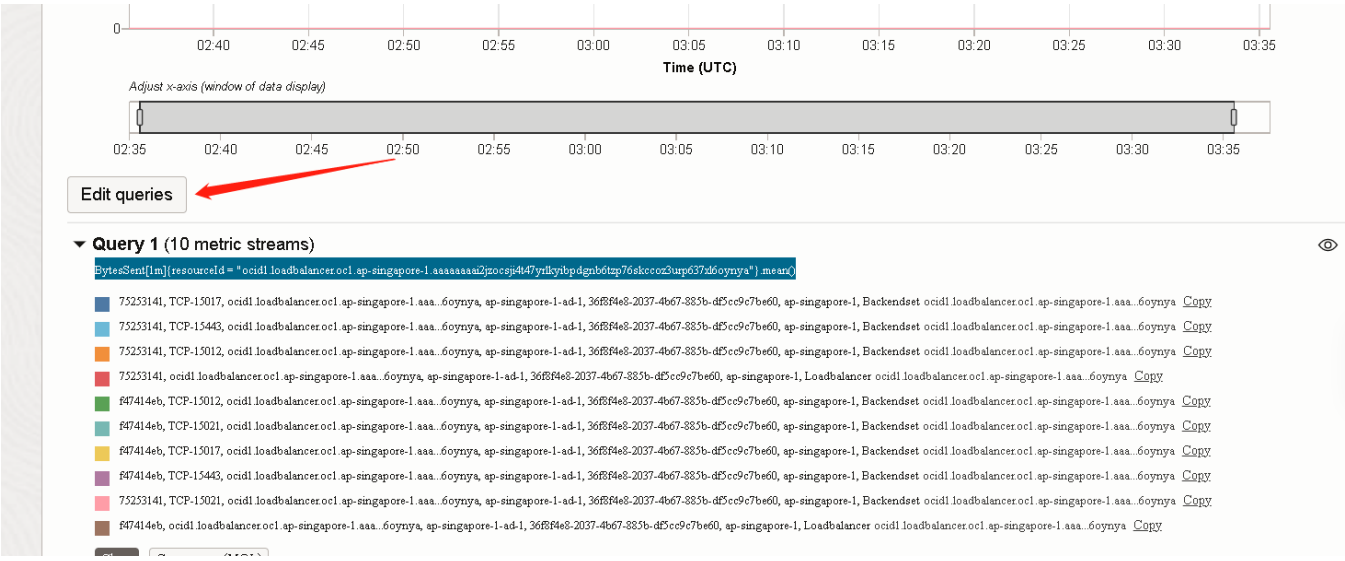


Select your target metric open explorer

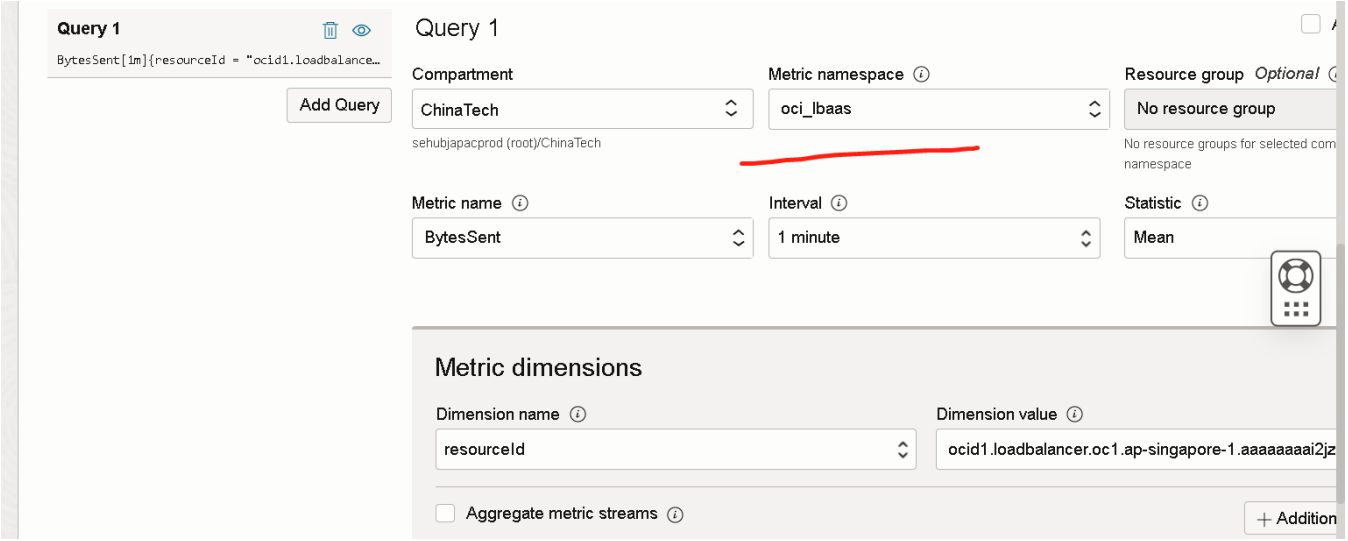


Copy MQL

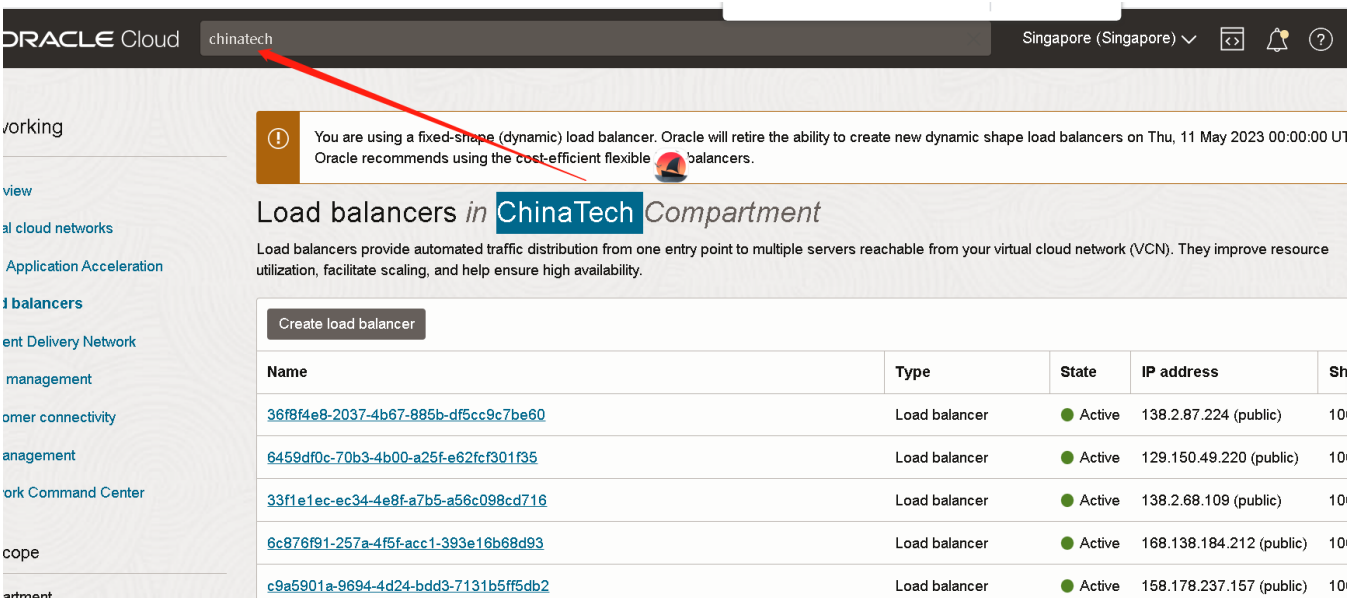




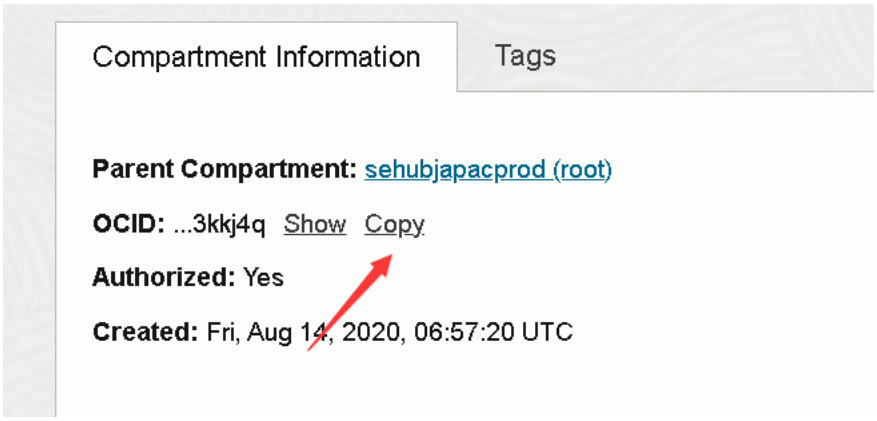
Here is the namespace



How to get compartmentId



Search the name in search bar



After you get all these, fill them in the metrics.yaml

The binary will try to export those metrics to be pulled by Prometheus

Oci_exporter Metric type

I have designed 2 types.

Single: if this MQL can only get 1 datapoint for one resource

Many: will get us many data points for many resources

Prometheus config

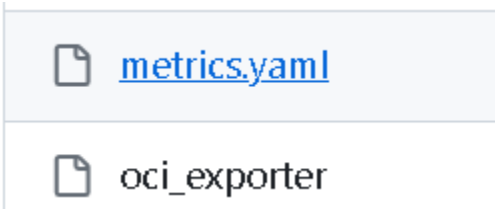
In order to let it discover metrics, we write config (prometheus.yaml)for it, yellow part is what we have added

```
global:
  scrape_interval:     15s # By default, scrape targets every 15 seconds.

  # Attach these labels to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: 'codelab-monitor'

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from
this config.
  - job_name: 'prometheus'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['localhost:9090']
  - job_name:       'oci-custom-job'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 60s

    static_configs:
      - targets: [  'localhost:8081']
        labels:
          group: 'production2'
```

Run OCI exporter

We need to run application against API key so we need to prepare key first.

In your application directory, we have a binary excutable and metrics.yaml file.



Oci_exporter.exe will listen on port 8080

When we access 127.1.1.1:8080/metrics

We can get metrics defined in metrics.yaml

We can add parameter to run on other port

Windows

oci_exporter.exe -listen-address=:8081 -config c:\adasdsad\config

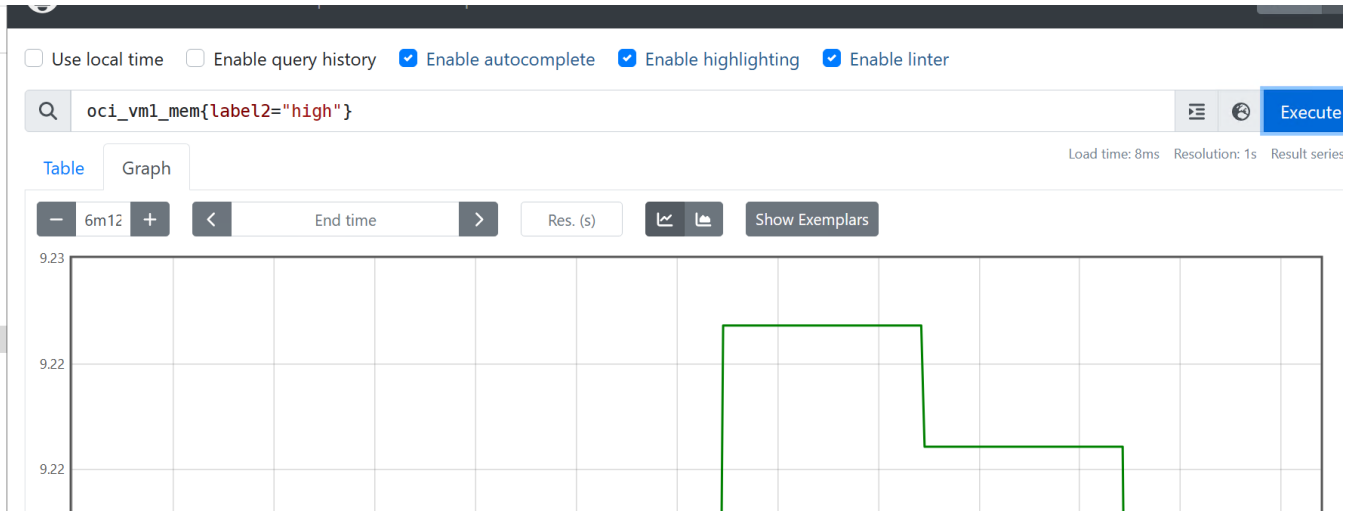
will listen on 8081

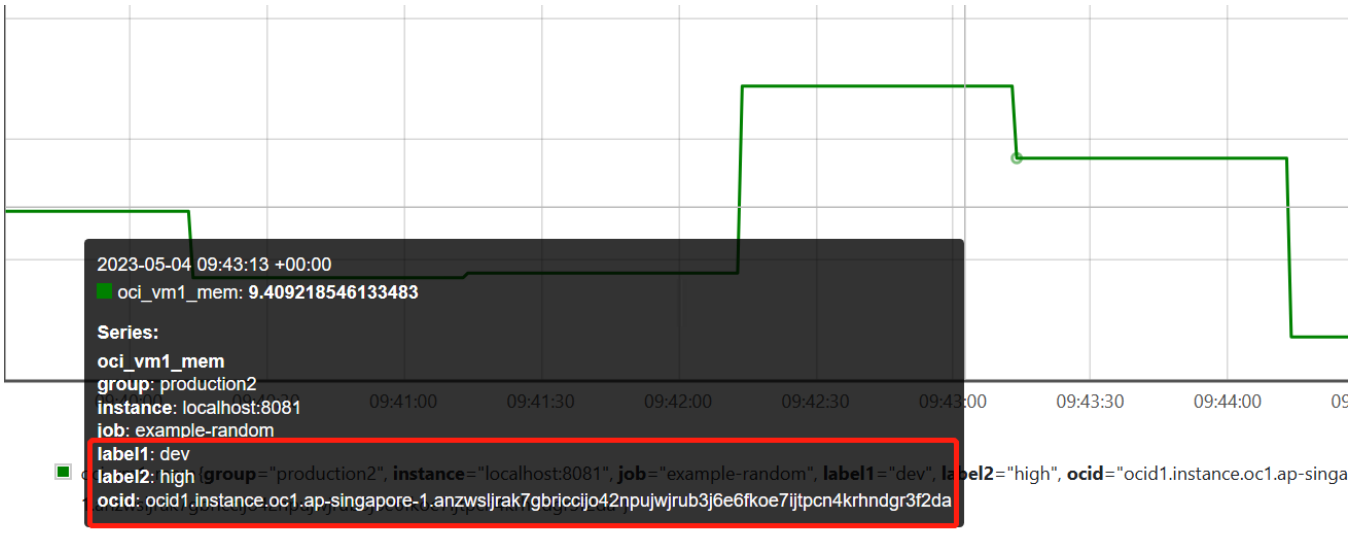
if oci_exporter.exe -listen-address=0.0.0.0:8081 -config c:\adasdsad\config

will bind all ips of local to listen on 8081

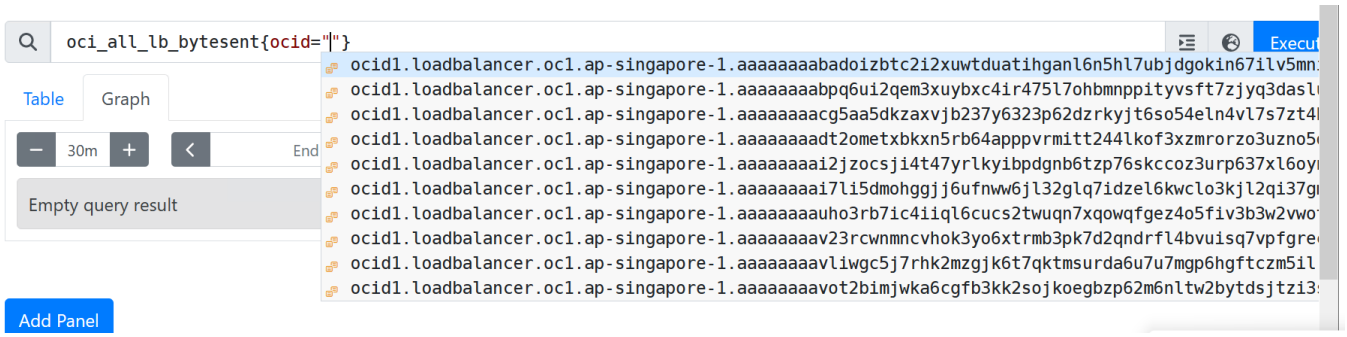
when we run Prometheus, we open dashboard

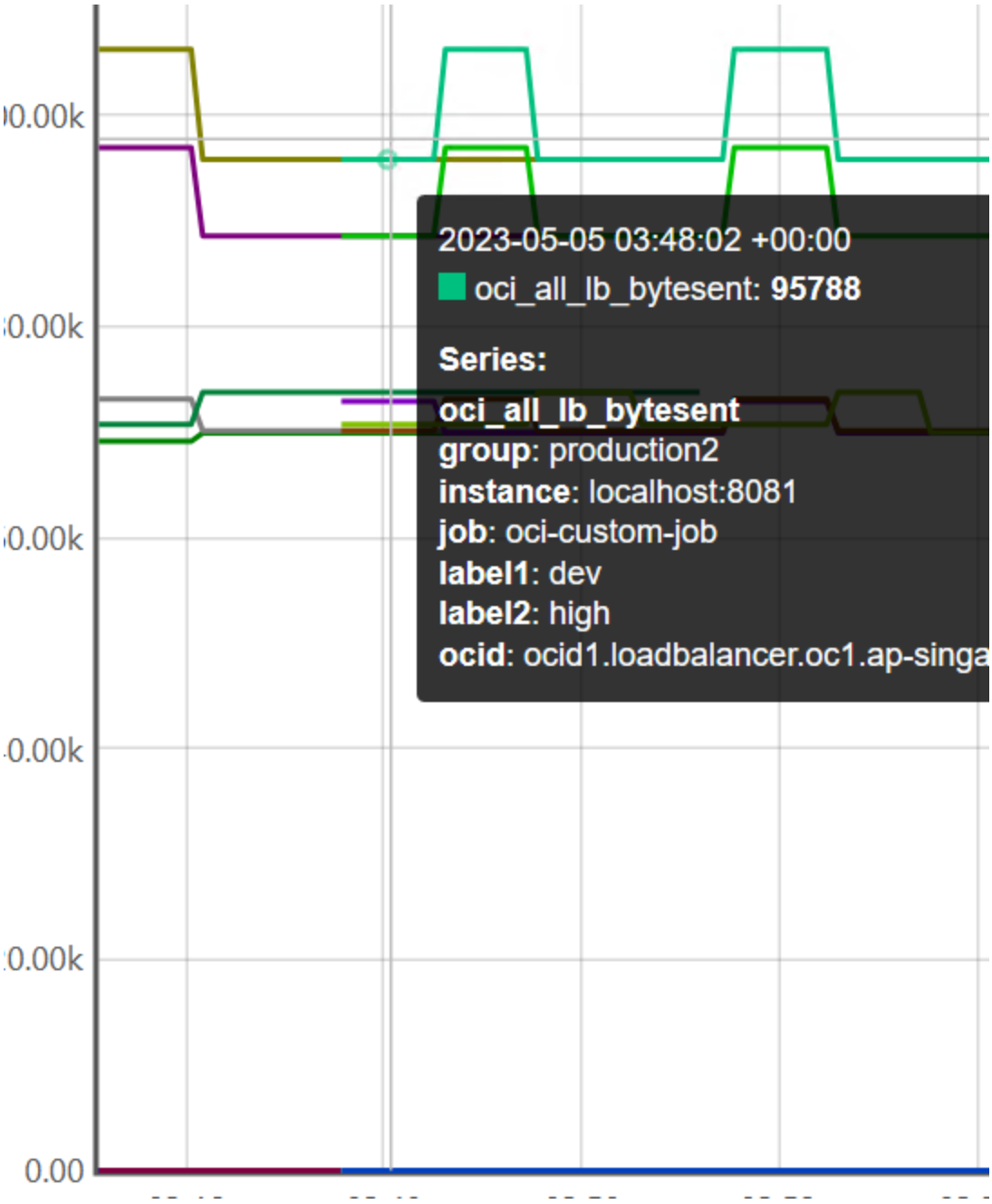
type the metric name and filter with the label





When we type promQl we get auto completions





Linux

./oci_exporter -config=/home/opc/.oci/config -listen-address=:8081

Docker

Dockerfile

```
FROM golang

# Set environment variables
```

```
# Update the package repository and install packages
RUN apt-get update && \
    apt-get install -y \
    iproute2 \
    wget \
    git \
    build-essential \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Set a working directory
WORKDIR /app

# Copy files into the container (if needed)
COPY . /app

RUN chmod +x /app/oci_exporter
```

build

```
docker build -t oci_exporter .
```

run

```
docker run  oci_exporter  /app/oci_exporter -config=/app/config
```

- /app/config is api key config file, need to mount volume into docker
- need to make sure key.pem in config file is existed.